CS65 Peerist Requirements Specification

Christopher Hoskins, Darius Moseley, Michael Chan, Jorge Manzo

CS 461

Fall 2019

Abstract

Peerist will be a web application that allows academic writers to receive feedback on their writing without worrying about plagiarism. This document discusses the features that are expected for Peerist as a minimum viable product. Also discussed are the features and functionality that will be provided by the client for Peerist (features that are not expected for us to implement). As this is mainly a front-end project, the UX and UI are incredibly important, which is also gone over in this document. A detailed general state diagram is included for the Peerist web application.

Contents

Introduction

This section contains agreed upon requirements specifications for the Peerist academic peer-review system, along with descriptions of the core functionality.

System Purpose

The aim of Peerist is to create a tool for collaboration in the space of academic research papers. There is a lack of tools available for researchers to collaborate and peer-review their work. There are document editing tools currently available, but they lack the specific tools and features required for academic papers. Currently the process of providing and sending feedback on academic papers is time-consuming. Peerist will provide an online platform to make it easier for users to provide feedback for academic works in a way that is modern and pleasant. This service will also enforce private peer-reviewing once segments are composed into larger components.

System Scope

The main end product that will be delivered is a functioning web application prototype. It will consist of a PostgresQL database, Hasura, GraphQL, and React.js. The PostgresQL database will be used to store user information such as individual user records, segments of writing and papers, as well as the relationship between users, their writings and their circles of trusted users on Peerist. GraphQL queries will be used to retrieve data from our PostgresQL database, allowing us to specify and request only the data we desire, thus being highly efficient. We will also produce React.js components as this is where most of the front-end work that we will be doing will be. Theses components will plug into the existing code base of Peerist and provide the functionality described in the System Purpose.

System Overview

System Context

As an academic writer, the user will use our application by first providing a segment of their writing. Upon uploading, the writer also specifies the subject their writing is related to (ex: Computer Science). The provided writing segment will be verified to be within the length requirement before being saved in the database. After saving, the user will be paired with other users of Peerist for their feedback. For the writer of the segment to be matched with peer reviewers, the other users must be within the same subject field.

Beginning the peer-review of the segment, the reviewers will be present with short prompts that will guide the review process. Each review user will answer the prompts, storing their responses within the database. The reviewing users can also make modifications and corrections to the segment, and these modification. will be saved as versions in the database, being tied to the user that made the change. This completes the review for the individual segment.

After the writer has enough segments reviewed, they can combine their segments on Peerist into one larger paper. The writer will then be paired with another set of peer-review users, however these will be limited to users approved by the writer. Once matched, the writer and the reviewing users will be provided the same prompts as a segment review to guide the process. Any changes made by the reviewers will be saved and tied to the authors of the change.

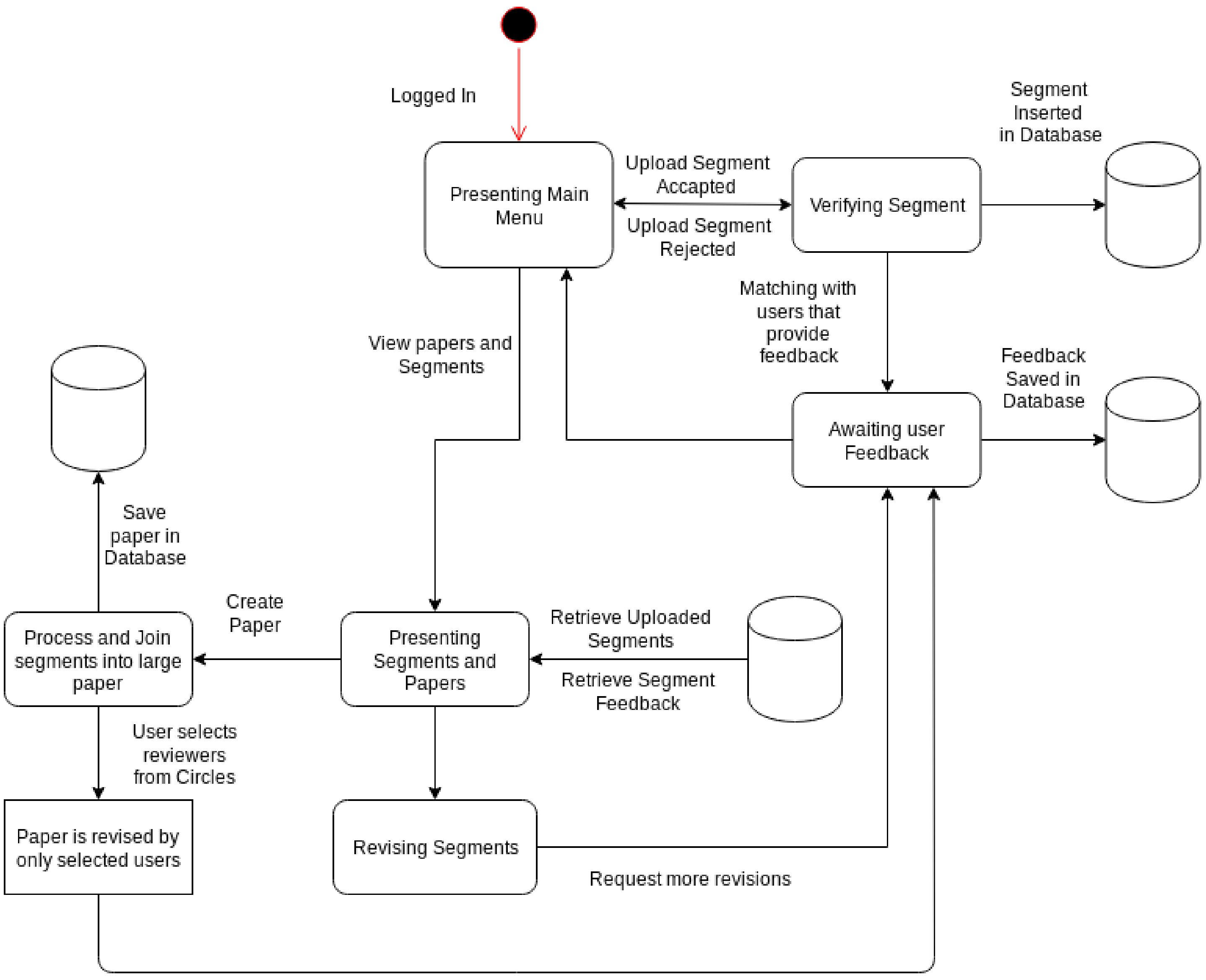

Logged In
Presenting Main Menu
Upload Segment Accapted
Upload Segment Rejected
Verifying Segment
Segment Inserted in Database
Matching with users that provide feedback
View papers and Segments
Awaiting user Feedback
Feedback Saved in Database
Save paper in Database
Create Paper
Process and Join segments into large paper
Presenting Segments and Papers
Retrieve Uploaded Segments
Retrieve Segment Feedback
User selects reviewers from Circles
Paper is revised by only selected users
Revising Segments
Request more revisions

Figure 1: General State Diagram for the Peerist application

System Functions

One of the major system functions of Peerist will be the user matching system. Our client has indicated that this matching algorithm responsible for the matching of writers to reviewers will be provided to us. The next major system function is accepting uploads of academic writings and having them inserted into the database as segment records upon approval. Another important function is being able to recognize which user is logged in order to display the correct content to the correct user. The application will retrieve records from the database for presenting what segments the user has provided, including papers, and what users are in their circles. Upon viewing a segment or paper, the application also retrieves records of the changes made to each of the user's segments.

User Characteristics

Users of the Peerist application will include those in academia. These users will likely be graduate students focusing in on at most two subjects, and their writings focusing on a single subject. These users will already be well educated likely holding some kind of undergraduate degree. The technical expertise may vary due to the wide range of research subjects. Some users may experience difficulty when trying to use the application if they have little experience with using web applications. These users may also be concerned about plagiarism and how this application will protect their writing.

DEFINITIONS

Segment

A limited body of text to be reviewed. Segments can be independently versioned, and grouped into larger works. Segments will have a maximum word count. Segments can be reviewed by Peers or Circles.

Paper

Papers are made of segments. Papers are independently versioned. Papers can only be reviewed by Circles.

Circle

Circles are groups of users that share a specific relationship. These could be members of a university's department, members of a project team, or users that have otherwise selected each other. It is recommended that members in a circle have at some point seen each other in person.

Peers

Peers are groups of people that share a more general connection than Circles. It is more likely that members within a peer group have not met each other. Peers can include people with similar jobs or research fields. Peers are selected by Peerist algorithms.

Feedback Prompt

The Feedback Prompt is the topic of review. Prompts are pointed and have accompanying possibilities for edits. E.g. the Feedback Prompt "Word Choice" would allow users to select predetermined responses on diction, jargon, and term comprehension.

Version

Each paper and segment will be independently versioned each time they are updated. The version stream will allow the user to view changes made over time and the review results for each version.

System Requirements

Functional Requirements

Core functional requirements for Peerist will include being able to update a PostgresQL database, retrieve records from the database, and use React.js components to provide the front end of our work. More specifically, the user should be able to create an account, login to their account, submit segments for review, review other's segments, revise their own segments (versioning), create circles (groups), and combine segments into papers.

User Interface / User Experience

Peerist will have an efficient, simple, and memorable user experience. Methods to quantify these qualities will be number of clicks to satisfy functional requirements, general accessibility, and user persona walkthroughs. The overall design guideline for Peersit will be minimalist design. Items that are not critical or useful on a page for the user experience will be removed or adapted. An example of the minimalist design philosophy can be found in the appendix under General UI.

System security

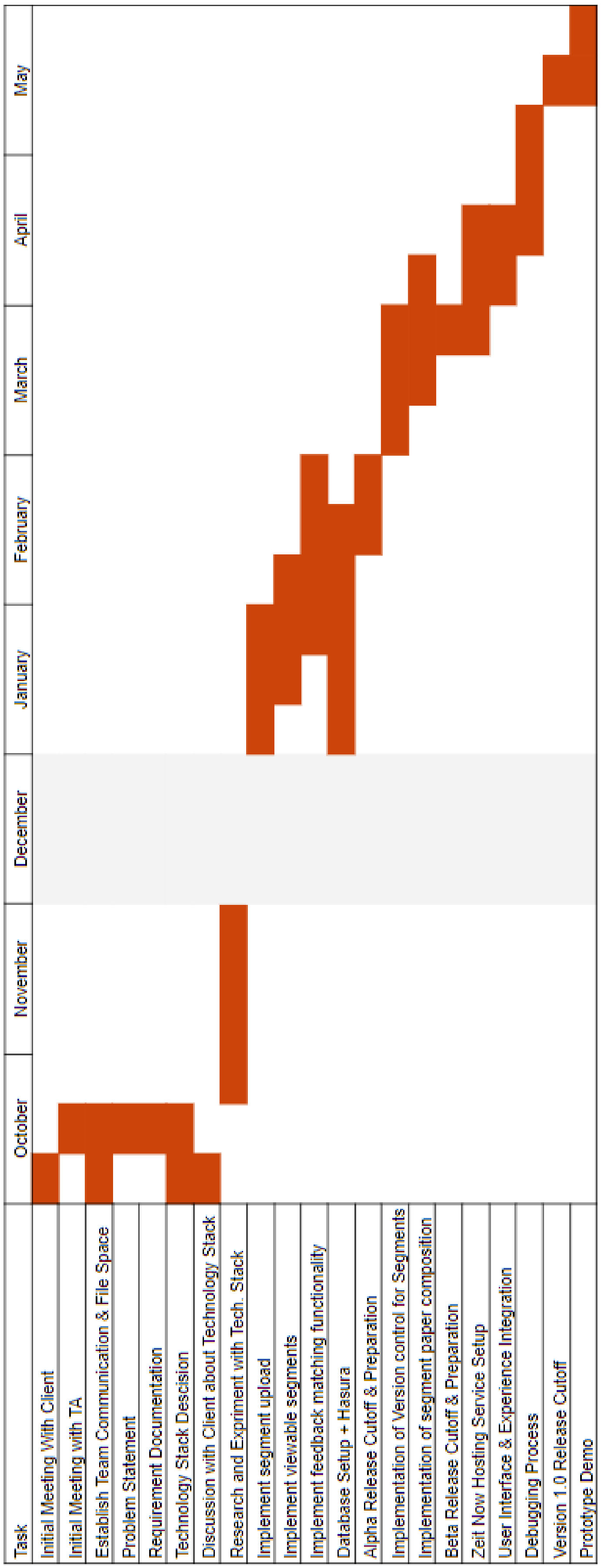
Peerist will utilize standard security features that are used by most websites today. Users will be able to login with their username and password, and their sessions will be maintained with cookies. Peerist will utilize HTTPS to prevent MITM (man-in-the-middle) attacks on login information and other sensitive user information. Additionally, passwords will be hashed before they're stored into the database.

Information Management

User ID and password will be securely stored onto the database. When users submit segments and papers, they're stored onto the database and linked to the user's ID. Additional versions will be stored with their corresponding segment and paper. Similarly, when users fill out the feedback prompt and send it to their recipient, feedback is directly stored with the recipient's segment, but the sender's ID remains associated to ensure that feedback is properly exchanged. Users will also be able to define their circle and have other users assigned to them as peers. The database will essentially have a table for each of the following: user information, segments, papers, and circles.

Appendix A

Gantt Chart


Task
October
November
December
January
February
March
April
May
Initial Meeting With Client
Initial Meeting with TA
Establish Team Communication & File Space
Problem Statement
Requirement Documentation
Technology Stack Descision
Discussion with Client about Technology Stack
Research and Expriment with Tech. Stack
Implement segment upload
Implement viewable segments
Implement feedback matching functionality
Database Setup + Hasura
Alpha Release Cutoff & Preparation
Implementation of Version control for Segments
Implementation of segment paper composition
Beta Release Cutoff & Preparation
Zeit Now Hosting Service Setup
User Interface & Experience Integration
Debugging Process
Version 1.0 Release Cutoff
Prototype Demo